BRAC UNIVERSITY

Inspiring Excellence

Department of Computer Science and Engineering

Course Code: CSE341	Credits: 1.5
Course Name: Microprocessors	Semester: Spring'25

Lab 02

Basic I/O, Advanced Arithmetic Operations and Flags

I. Topic Overview:

Instructions used to communicate with peripherals are called interrupts. There are many interrupts each dealing with a particular task. Students will familiarize themselves with the basic input output mechanisms of the assembly language. They will be looking into one of the most common interrupts, INT 21H and inquire how this very interrupt can be used for single key input, single key output and multiple characters output. Additionally, they'll perform some arithmetic operations upon taking inputs from the user.

II. Lesson Fit:

There is a prerequisite to this lab. Students must have a basic idea on the following concepts:

a. Registers
b. Some basic operations such as MOV, ADD, SUB, MUL and DIV
c. Character encoding using ASCII

III. Learning Outcome:

After this lecture, the students will be able to:

a. Perform I/O operations to make their assembly programs more interactive
b. Write more dynamic assembly programs
c. Familiarize themselves with how flag register works

IV. Anticipated Challenges and Possible Solutions

a. Students may find it difficult to map ASCII values to decimal/hex
Solutions:
i. Looking at the **ASCII** table may help

V. Acceptance and Evaluation

Students will show their progress as they complete each problem. They will be marked according to their class performance. There may be students who might not be able to finish all the tasks, they will submit them later and give a viva to get their performance mark. A deduction of 30% marks is applicable for late submission. The marks distribution is as follows:

Code: 50%
Viva: 50%

VI. Activity Detail:

a. Hour: 1

Discussing: Basic Input and Output

In order to carry out basic I/O operation, function specific values need to be kept in AH register before calling INT 21h which is used for a system call. The function dispatcher maps the value to carry out specific functions.

SN	Register	Value (decimal)	Used For
1	AH	1	Single Character Input
2	AH	2	Single Character Output
3	AH	9	String Output

Note: INT 21H call changes AX/AL. So if we have any value stored in AL we have to save it somewhere else before calling INT 21H

i) Single Character Input

We start by moving the decimal 1 to AH register and call Interrupt 21H. Upon running the program will wait for the user to give an input (Any Single character). The input will be stored in AL register

`MOV AH, 1` `INT 21H`	User given input will be saved in the AL register as **ASCII** value. To do arithmetic operations from a numeric input, an ascii to numeric conversion is needed.

ii) Single Character Output

The character to be printed must be in the DL register. Then we move decimal 2 to the AH register and call Interrupt 21H

`MOV DL,'A'`
`MOV AH, 2`
`INT 21H`
	The ASCII value of the desired output should be in the DL register. Here by moving A to DL we are moving the ascii value of A (65) to DL register.
	MOV DL, 65 would also print A

iii) String Output

A string is an array of characters. To show a string output we must define the string in the data segment.

`.DATA`
`course db "CSE341 $"`
`message db " is very fun! $"`
	$ = This dollar sign is called a String Terminator. It denotes the end of the string which is needed.

The values of variables and arrays are stored in the data segment of the memory. In order to fetch them we need the offset address (offset + segment). To print a string we need to store 9 in AH. While 9 is in AH, INT 21H expects DX to hold the offset address of the array/variable.

LEA is the name of the instruction that provides the offset address. LEA stands for Load Effective Address. The syntax for using LEA is:

LEA destination, source.

Source is the memory address (the array or the variable) whose offset address will be saved in the destination. The destination is a general register.

`LEA DX, course`
`MOV AH, 9`
`INT 21H`
	This will print **CSE341** on the emulator screen. Notice that if we do this multiple times the output is shown on the same line. It does not automatically go to the next line.

Printing on New Line:
We must move the cursor to the next line by executing Line Feed and Carriage Return. Like every other single character, we have to print their corresponding ascii values.
Carriage Return (cret) = 10d or 0DH
Line Feed (newl) = 13d or 0AH

Full Example:

```
.MODEL SMALL
.STACK 100H
.DATA
ask db "Enter a digit: $"
result db "My favourite digit is: $"

.CODE
MOV AX,@DATA
MOV DS,AX
MOV ES, AX

;Showing first message
LEA DX, ask
MOV AH, 9
INT 21H
;Taking Input
MOV AH, 1
INT 21H
MOV BL, AL
```

```
;Moving to a New Line
MOV AH, 2
MOV DL, 10
INT 21H
MOV AH, 2
MOV DL, 13
INT 21H

;Showing Result
LEA DX, result
MOV AH, 9
INT 21H
MOV AH, 2
MOV DL, BL
INT 21H

MOV AX,4C00H
INT 21H

MAIN ENDP
    END MAIN
```

Problem Task: Task 01-04(Page 6-7).

b. Hour: 2

Discussion:
Check the problem tasks while the students continue with the rest.
Problem Task: Task 05-09(Page 7-8).

c. Hour: 3

Discussion: Flag register

There is one special type of register called the flag register. It contains flags. Flags are bits of information which indicate the state of the processor after an arithmetic operation. Decisions are made by the processor based on the flags. **Flags are of two types**:

1. **Status Flags**: The processor uses these flags to reflect the result of an operation.
2. **Control Flags**: These are used to enable/disable certain operations of the CPU.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
				OF	DF	IF	TF	SF	ZF		AF		PF		CF

Bit number **10**, **9** and **8** are control flags and the rest **non-empty** bits are **status flags**. The empty bits have no significance.

Status Flags

Carry Flag (CF): CF = 1 when there is a carry out from the MSB during addition or a borrow into the MSB during subtraction.

Parity Flag (PF): PF = 1 when the low byte of a result of an operation contains an even number of ones.

Auxiliary Flag (AX): AX = 1 when there is a carry out from bit number 3 to bit number 4 in addition or borrow into bit number 3 during subtraction. (**Note that the bit number starts from 0**).

Zero Flag (ZF): ZF = 1, if the result of an operation in 0.

Sign Flag (SF): SF = 1 when the MSB of a signed number is 1.

Overflow Flag (OF): OF = 1 when there is an overflow. There are 3 types of overflows:- Signed Only, Unsigned Only and Both. To understand overflow, you must know by now that 1111b has 2 different meanings in the context of unsigned and signed numbers. If you are not clear about this please read the DLD book.
An example of unsigned overflow where the maximum number of bits is 4,

1111 + 1000 = 10111

Here the MSB gets discarded and thus data loss. As an example of signed overflow where the maximum range is 8 bits. For signed numbers the MSB represents the sign. 1 for negative and 0 for positive.

0111 1111 + 0111 1111 = 1111 1110

Both the operands start with 0 therefore both are positive numbers. Addition of 2 positive numbers have to be positive. But if we look at the answer, it starts with 1 therefore the number is a negative number.

How instructions affect flags:

Instructions	**Affected Flags**
MOV	None
ADD/SUB	All
INC/DEC	All (Except CF)
NEG	All (CF =1; Unless result is 0 and OF= 1, if the operand is 8000H and 80H)

It has been previously mentioned that the processor takes decisions based on the flags. In this section we will see how these decisions are made. In JAVA, decisions are made using **"if"** conditions. If the condition is satisfied then a portion of code runs and if not, the program **jumps** to another section of code. In assembly this whole system is done by two operations- **compare(CMP)** and **jump(JMP)**. The jump can be condition dependent and also independent.
Problem Task: Task 10-12(Page 8).

VII. Home Tasks (All the unfinished lab tasks)

Lab 2 Activity List

Task 01
Take a character input and display it. Display the message "Please insert a character: " before taking an input.

Task 02
Perform addition/subtraction/division/multiplication by taking inputs from the user.
Note: Display appropriate messages when taking input and showing the output.

Task 03
Write instructions to do the following.
a. Read a character, and display it at the next position on the same line.

b. Read an uppercase letter (omit error checking), and display it at the next position on the same line in lower case.

Task 04

Read an uppercase letter (omit error checking), and display it at the next position on the next line in lower case.

Task 05

Write a program to:
(a) display a "?"
(b) read two decimal digits whose sum is less than 10,
(c) display them and their sum on the next line, with an appropriate message.

Sample execution:

?27
THE SUM OF 2 AND 7 IS 9

Task 06

Write a program to:
(a) prompt the user, (b) read first, middle, and last initials of a person's name, and then display them down the left margin.

Sample execution:

ENTER THREE INITIALS: JFK
J
F
K

Task 07

Write a program to read one of the hex digits A-F, and display it on the next line in decimal.

Sample execution:

ENTER A HEX DIGIT: C
IN DECIMAL IT IS 12

Task 08

Write a program to display a 10 x 10 solid box of asterisks.
HINT: Declare a string in the data segment that specifies the box, and display it with INT 21h, function 9h

Task 09

Write a program that takes two multi digit decimal numbers as input and performs ADD operations between them.

Task 10

ADD AL, BL. Where AL contains 80h, BL contains 80h. Identify the status of different status flags.

Task 11

Suppose that AX and BX both contain positive numbers and ADD AX, BX is executed. Show that there is a carry into the MSB but no carry out of the MSB if, and only if, signed over flow occurs.

Task 12

Suppose AX and BX both contain negative numbers, and ADD AX,BX is executed. Show that there is a carry out of the MSB but no carry into the MSB if, and only if, signed overflow occurs.